Revision History

Date	Version	Description	Author
13.11.2023	1.0	Design of the program.	Elif Özmen

TABLE OF CONTENTS

1 Introduction

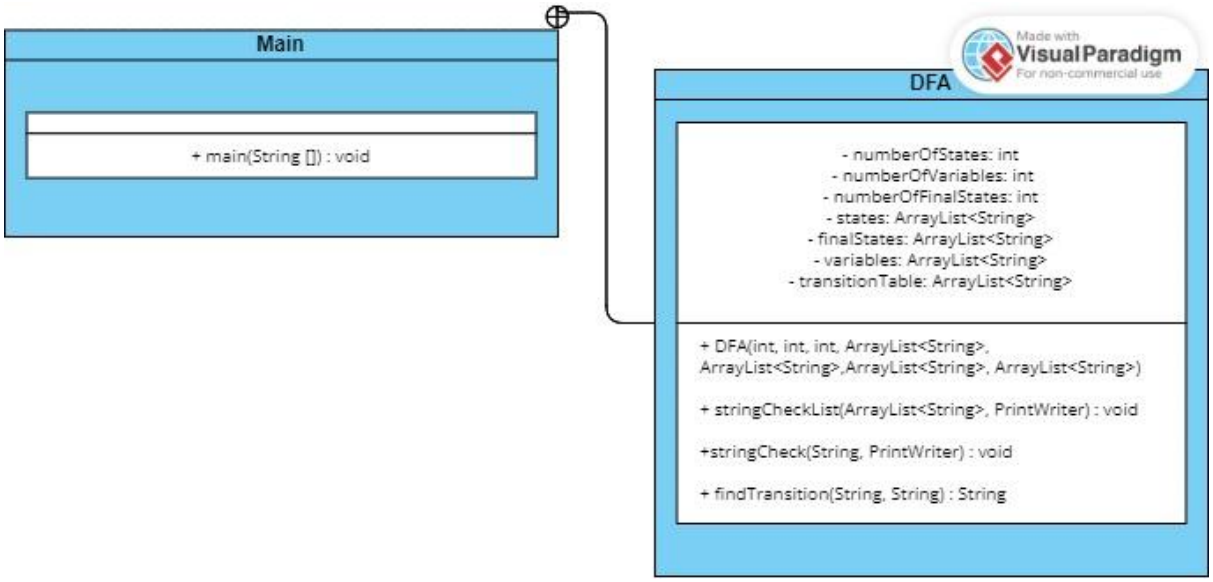
In this report I mainly tried to explain how I implement the assignment given in my CS410 class. Report mainly focuses on the introduction to the problem, design considerations, algorithms and data structures used in implementation, input/output format, error handling, testing, code structure and challenges faced during implementation.
The problem in this project was simulating a Deterministic Finite Automaton (DFA) which works with any given alphabet, number of states, number of variables and transitions. The simulation has to work correctly as a DFA would do and it should read the information from the given input file then print the output information about the given string, states visited, string being accepted or rejected.

1.1 References

1.1.1 Project References

#	Document Identifier	Document Title
[SRS]	DFA-SRS-1	DFA Software Requirements Specifications

2 Software Architecture overview



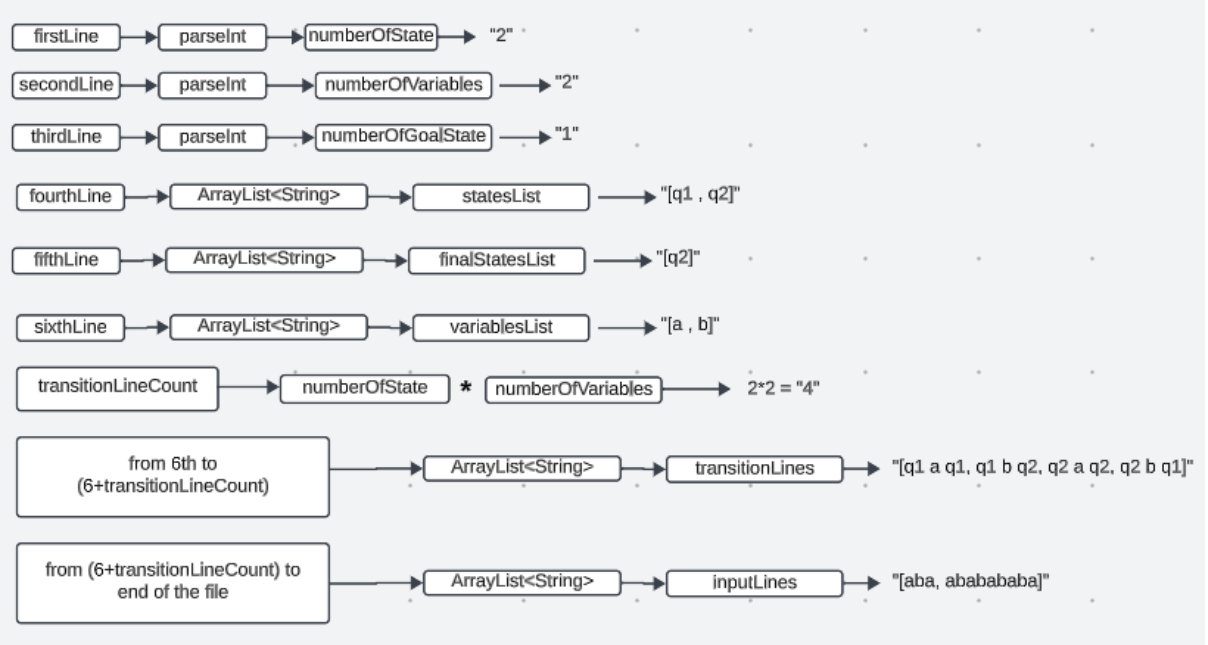
3 Software design description

3.1.1 Component interfaces

Input Data: Input is taken from the text file constructed with the same way with the example input file given. Input file's name is taken from the user in the console.
Output Data: In the program the results are written in a output file and also to the console. Output file's name is taken from the user in the console.

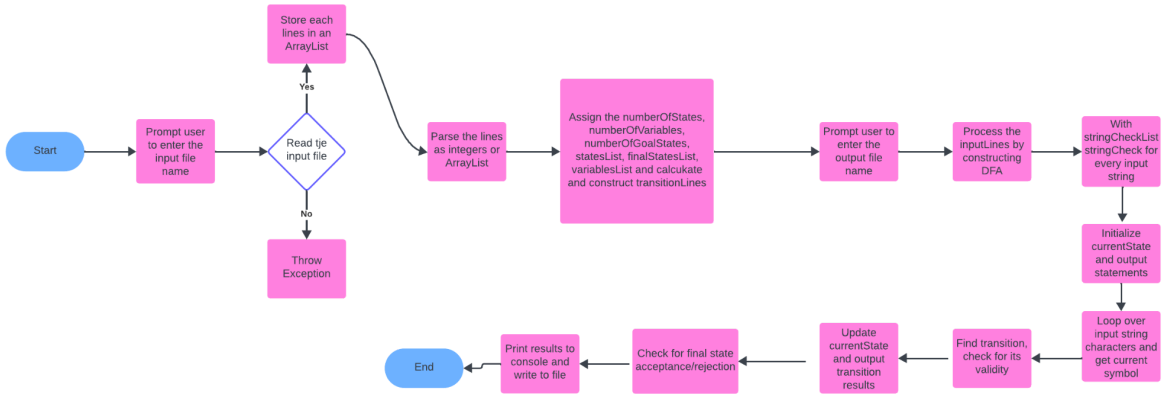
3.1.2 Designing Description

Before starting to implement, I considered how to read the information from the input file. I stored each line in an ArrayList. In the given example input file it shows that the first line is the number of states, second line is the number of variables which means language, third line is the number of final states in DFA. For the first three lines, since they are the count of the elements, I decided to parse them to integer and make each one integer parameters. For the fourth line of the input file, it is the information about the states names. I decided to keep them in a ArrayList of strings, each state name is one element in the list. As it is similar in the fifth and the sixth lines which are the names of final states and the symbols, I applied the same process to them. It is always precise, the number of the first six lines in the input file and what information they contain. However, it starts to move to transition lines on line seven, and this can change to any number depending on the number of states and variables.The transition lines come from the transition table's every row and column. As I see it's count equals to product of number of state and number of variables (# of state * # of variables, I assigned this number to an integer parameter called transitionLineCount, and I constructed a for loop to read the lines from the input file starting with the 7th line and continuing until the 6+transitionLineCount line. I put them into a ArrayList of Strings that each transition line is an element of the list.
After the transition lines end, in the input file, there starts the string lines that will be taken to check if it is accepted or not in the DFA. So it starts with the (6 + transitionLineCount) line and continues until the input file ends. Again each string is an element of the ArrayList. When I was making these implementations I had to visualize these to make them easier to implement. So I drew the visual in the below l and it helped me very much while implementing.



After reading the input file and retrieving the information, I had to construct the DFA. So I created a class called DFA and I created a constructor for this class with the parameters I took from the input file. In DFA class there are three methods. “stringCheck” method is for analyzing each input string and deciding if the DFA will accept it or reject it. It implements the basic process of traveling through states respective with the input symbols and determining whether the ultimate state has been reached. I implemented a stringCheckList function to repeatedly apply the stringCheck method to each input string in the form of a list of strings as I implemented the input string lines as an ArrayList. The findTransition method iterates through each entry in the DFA's transitionLines which represents the transitions between states based on input symbols. After I implemented these I called the DFA constructor in the main method. I decided to print the results to both the console and to an output file. I decided to use Scanner because I wanted to take the input file’s name from the user to make it more flexible. Also I decided to take the output file’s name from the user, again with the same reason. So, this was my decision and implementation phase, after I finished I tried some test cases and made some improvements. I created different input files with the DFA’s I constructed by myself and I added it to the project file also.

3.1.3 Workflows and algorithms



In the implementation the stringCheck method implements the main algorithm for analyzing input strings. This algorithm iterates through each symbol in the input string, updating the current state of the DFA based on transitions defined in the transitionLines and it uses a linear search approach. Also similar to that findTransition method uses similar approach because it iterates through transitionLines, locates the specific transition according to current state and input symbol. With this process it dynamically determines the next state based on the transition table (transitionLines).

In terms of data structures, in the implementation there are so many ArrayLists. The states, finalStates, variables, and transitionLines are stored in ArrayLists which provides dynamic storage that can work with every element count.

The constructor of the DFA class initializes these data structures with the parameters retrieved from the input file. With this, it makes the code work with Object Oriented Programming principles.

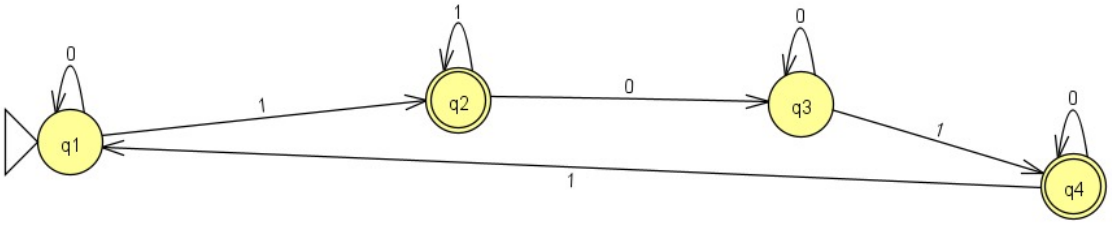
4 COTS Identification

COTS (commercial of the shelf) libraries used in DFA are the following:

- java.io.*;
- java.util.ArrayList;
- java.util.Arrays;
- java.util.Scanner;

5 Testing and Error Handling

To test whether my program was working correctly, I created DFAs myself and wrote their input files to try, along with the sample input file provided. In general it was working correctly but there were some problems in printing it to the console and writing it to the output file so I fixed them, too.
Here is the DFA I constructed as a test case:



For the error handling I used the IDE's tools. I have three exceptions: IOException, FileNotFoundException and NumberFormatException.

6 Challenges Faced

To me, writing this report was harder than implementing it because it is harder to visualize the designs, patterns and algorithms you designed in your mind while coding. Other challenges I faced while coding was mostly where I was finding the logic while I was trying to implement algorithms. The most challenging thing about this project is it has to be a dynamic program that can work with every input correctly.